Universidade Federal de Uberlândia

Bacharelado em sistemas de informação

Faculdade de computação

Organização e recuperação da informação

Atividade prática TP2

Uberlândia – MG

Universidade Federal de Uberlândia

Bacharelado em sistemas de informação

Faculdade de computação

Maycon douglasBatista dos santos11921BSI209

Uberlândia – MG

Relatório

Documento de relatório para o TP2 referentes aos modelos booleanos e tf*idf mostrando os resultados obtidos durantes a prática de **organização e recuperação da informação** na disciplina curso **Bacharelado em Sistemas de Informação**.

Professor: Rodrigo Sanches Miani

Sumário

Objetivo

O objetivo desta aula é entender revisar o modelo booleano aplicando o conceito de frequência de termos e a partir desse entendimento poder aplicar conceitos mais avançados de recuperação da informação que são usados até hoje.

Exercício 1)

Criar um programa que faça o seguinte:

1) Leia múltiplos arquivos de texto (um diretório, por exemplo)

```
6   # ler todos os documentos de uma pasta com a extenção indicada e
7   # salva na pasta de destino indicada
8   def bagOfWords2 (source = './docs', ext = '.txt' , dest = None):
9
10      bag = []
11      docs = []
12
13      for dir, subPastas, arquivos in os.walk(source):
14
15          for arquivo in arquivos:
16
17              if Regex.findall(ext + '$', arquivo):
18
19                  arg = open(source + '/' + arquivo, 'r')
20
21                  arg = uteis.normaliza(arg.read())
22
23                  if(dest != None):
24                      uteis.saveFile(arg, dest + '/' + arquivo)
25
26                  docs.append(arquivo)
27                  bag.append(arg)
28
29      return bag, docs
```

2) Crie o vocabulário (termos de indexação únicos) dos arquivos de texto removendo qualquer tipo de pontuação e considerando somente letras minúsculas

```
36
37  def normaliza(words):
38
39      bag = []
40
41      palavras = words.replace('\n', ' ').replace(',','').split(' ')
42
43      for item in palavras:
44          elem = unidecode.unidecode(item.lower())
45
46          if elem not in bag and elem != '':
47              bag.append(elem)
48
49      bag.sort()
50      return bag
51
```

3) Grave o vocabulário em arquivo texto

```python
# recebe as lista de palavras ja normalizada e grava im arquivo
#  com extenção .index

def saveFile (bagOfWords, name):

    arg = open(name + '.index','w')

    for item in bagOfWords:
        arg.write(item + '\n')


```

4) Leia o vocabulário em arquivo texto e exiba no console a BoW dos documentos (ausência/presença do termo em um determinado documento)

```python
# ler todos os docs de uma pasta e salva a bag of words da cada um em uma pasta index
if __name__ == '__main__':

    source = input('Entre com o caminho relativo dos arquivos que serão indexados: ')

    ext = input('Qual extenção dos arquivos que serão indexados: ')

    dest = input('Entre com o caminho relátivo de onde serão guardados: ')

    listaBOW, namesFiles = bagOfWords2 ( source , ext, dest)

    name = input('Agora entre com o arquivo com a busca: ')

    keyWords = uteis.normaliza(open(name).read())

    result = matrixBollean(listaBOW, keyWords)

    exibirBOW(result, namesFiles, keyWords)

```

```
maycon@maycon-Inspiron-3584:~/5Periodo/ORI/TP2$ python3 Ex2/ex2.py
entre com nome do arquivo vocabulário: keywords.txt
entre com nome/caminho do diretório: docs
['to', 'do', 'is', 'be', 'or', 'not', 'i', 'am', 'what', 'think', 'therefore', 'da', 'let', 'it']
[0.0, 1.073, 0.0, 0.0, 0.0, 0.0, 2.0, 1.0, 0.0, 2.0, 2.0, 0.0, 0.0, 0.0] -> doc3.txt
[3.0, 0.83, 4.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0] -> doc1.txt
[2.0, 0.0, 0.0, 0.0, 2.0, 2.0, 2.0, 2.0, 2.0, 0.0, 0.0, 0.0, 0.0, 0.0] -> doc2.txt
[0.0, 1.073, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 5.17, 4.0, 4.0] -> doc4.txt
maycon@maycon-Inspiron-3584:~/5Periodo/ORI/TP2$ ls
docs  Ex1  Ex2  Ex3  hinos  index  keywords.txt  relatorio.odt
maycon@maycon-Inspiron-3584:~/5Periodo/ORI/TP2$
```

Exercício 2

Criar um programa que receba como parâmetro um arquivo com o vocabulário e um diretório ou caminho com os documentos (arquivos de texto). O programa, então, deverá:

1) Calcular e exibir no console o TF-IDF de cada documento. Testar com a coleção de quatro docuentos do livro/sala de aula. Verificar se os valores estão corretos.

```python
# calcula a frequencia de uma palavra na lista passasa como primeiro
# parametro (vocalulario indexado) na lista do segundo parametro (doc indexado)
def frequencia (keyWords, lista):

    result = []
    for key in keyWords:

        result.append(lista.count(key))

    return result
```

```python


# calcula a frequencia da lista de vocabulario (segundo parametro) na matrix
# que seria o nossa bag (segundo parametro) usando a função descrita acima
def TF (bag, keyWords):

    result = []

    for lista in bag:

        result.append(frequencia(keyWords, lista))

    return result
```

```python
# recebe um TF e converte para sua variação
def TFvar (tf):

    result = []
    for doc in tf:
        linha = []
        for item in doc:

            if(item > 0):
                aux = 1 + math.log2(item)
            else:
                aux = 0

            linha.append(aux)
        result.append(linha)

    return result
```

```python
# calcula o idf para cara termo (coluna) na minha matriz de frequencia
def IDF (tf):

    idf = []
    for i in range(len(tf[0])):

        idf.append(0)

    for i in range(len(tf)):
        for j in range(len(tf[i])):

            if(tf[i][j] > 0):
                idf[j] += 1

    for i in range(len(idf)):
        if idf[i] > 0:
            idf[i] = math.log2(len(tf) / idf[i])

    return idf
```

```python
# pega cada valor da matris e multiplica pelo seu valor idf
def TF_IDF (tf, idf):

    result = []
    for lin in range(len(tf)):

        linha = []
        for col in range(len(tf[lin])):

            linha.append(round(float(tf[lin][col]) * float(idf[col]), 3))

        result.append(linha)

    return result
```

```python
if __name__ == '__main__':

    keyWords = open(input("entre com nome do arquivo vocabulário: ")).read()

    keyWords = normaliza(keyWords, ord=False)

    nomeDir = input("entre com nome/caminho do diretório: ")

    bag, nomes = bagOfWords(source=nomeDir)

    tf = TF(bag, keyWords)

    tf = TFvar(tf)

    idf = IDF(tf)

    final = TF_IDF(tf, idf)

    print(keyWords)
    for i in range(len(final)):

        print(f'{final[i]} -> {nomes[i]}')

```

Resultado:

```
maycon@maycon-Inspiron-3584:~/5Periodo/ORI/TP2$ ls
docs  Ex1  Ex2  Ex3  hinos  index  keywords.txt  relatorio.odt
maycon@maycon-Inspiron-3584:~/5Periodo/ORI/TP2$ python3 Ex2/ex2.py
entre com nome do arquivo vocabulário: keywords.txt
entre com nome/caminho do diretório: docs
['to', 'do', 'is', 'be', 'or', 'not', 'i', 'am', 'what', 'think', 'therefore', 'da', 'let', 'it']
[0.0, 1.073, 0.0, 0.0, 0.0, 0.0, 2.0, 1.0, 0.0, 2.0, 2.0, 0.0, 0.0, 0.0] -> doc3.txt
[3.0, 0.83, 4.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0] -> doc1.txt
[2.0, 0.0, 0.0, 0.0, 2.0, 2.0, 2.0, 2.0, 2.0, 0.0, 0.0, 0.0, 0.0, 0.0] -> doc2.txt
[0.0, 1.073, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 5.17, 4.0, 4.0] -> doc4.txt
maycon@maycon-Inspiron-3584:~/5Periodo/ORI/TP2$
```

Exercício 3

1) Crie uma coleção de documentos com os hinos dos clubes dos 20 times que irão jogar a Série A do Campeonato Brasileiro de 2023.

```
maycon@maycon-Inspiron-3584:~/5Periodo/ORI/TP2$ ls hinos/
'Hino do Botafogo.txt'                  'Hino do Goiás Esporte Clube.txt'                'Hino Oficial do Clube Atlético Bragantino.txt'
'Hino do Corinthians.txt'               'Hino do Grêmio.txt'                             'Hino Oficial Do Clube Atlético Mineiro.txt'
'Hino do Cruzeiro.txt'                  'Hino do Internacional.txt'                      'Hino Oficial do Coritiba Foot-ball Club.txt'
'Hino do Cuiabá Esporte Clube (MT).txt' 'Hino do Palmeiras.txt'                          'Hino Oficial do Esporte Clube Bahia.txt'
'Hino do Flamengo.txt'                  'Hino do Vasco da Gama.txt'                      'Hino Oficial Do Santos Futebol Clube.txt'
'Hino do Fluminense.txt'                'Hino Oficial do América Futebol Clube (MG).txt' 'Hino Oficial do São Paulo Futebol Clube.txt'
'Hino do Fortaleza Esporte Clube.txt'   'Hino Oficial do Club Athletico Paranaense.txt'
maycon@maycon-Inspiron-3584:~/5Periodo/ORI/TP2$
```

2) Use o programa criado no Exercício 2 para responder as seguintes perguntas:

Código:

```python
# recebe um doc (td*idf) valor do menor peso e nomes dos arquivos lidos
# considerando que o doc e o nomes tem o mesmo tamanho e a
# posição i nos nomes indicada se refere a no doc pelo valor (tf*idf)
def menorPeso (doc, valor, nomes):

    lista = []
    for i in range(len(doc)):

        if valor == doc[i]:
            lista.append(nomes[i])
    return lista

```

```python
# apresentação do exe 3
if __name__ == '__main__':

    keyWords = open(input("entre com nome do arquivo vocabulário: ")).read()

    nomeDir = input("entre com nome/caminho do diretório: ")

    bag, nomes, voc = uteis.Ex2(keyWords, nomeDir)

    print(f'O tamaho do vocbulário é : {len(voc)}')

    print(f'vocabulário: {voc}')
    for nome in nomes:
        print(f'doc {nome} tf*idf é :')

        for doc in bag:
            print(doc)

            menor = min(doc)

            lista = menorPeso(doc, menor, voc)

            print(f'menor valor {menor} com os termos {lista}')

```

a) Qual o tamanho do vocabulário da coleção?
Linha 26
b) Encontre o TF-IDF de cada um dos documentos da coleção.
Linha 24
c) Qual termo(s), em qual documento, possui o maior peso TF-IDF?
Linhas 37, 39

Resultado:

```
maycon@maycon-Inspiron-3584:~/5Periodo/ORI/TP2$ ls
docs  Ex1  Ex2  Ex3  hinos  index  keywords.txt  relatorio.odt  vocab3.txt
maycon@maycon-Inspiron-3584:~/5Periodo/ORI/TP2$ python3 Ex3/ex3.py
entre com nome do arquivo vocabulário: vocab3.txt
entre com nome/caminho do diretório: docs
O tamaho do vocbulário é : 20
vocabulário: ['dourado', 'baixada', 'rasqueado', 'cuiaba', 'cruzeiro', 'fortaleza', 'clube', 'de', 'gloria', 'e', 'tradicao', 'salve', 'o', 'tr
icolor', 'paulista', 'athletico!', 'mantendo', 'nosso', 'espirito', 'esportivo']
doc doc3.txt tf*idf é :
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
menor valor 0.0 com os termos ['dourado', 'baixada', 'rasqueado', 'cuiaba', 'cruzeiro', 'fortaleza', 'clube', 'de', 'gloria', 'e', 'tradicao',
'salve', 'o', 'tricolor', 'paulista', 'athletico!', 'mantendo', 'nosso', 'espirito', 'esportivo']
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
menor valor 0.0 com os termos ['dourado', 'baixada', 'rasqueado', 'cuiaba', 'cruzeiro', 'fortaleza', 'clube', 'de', 'gloria', 'e', 'tradicao',
'salve', 'o', 'tricolor', 'paulista', 'athletico!', 'mantendo', 'nosso', 'espirito', 'esportivo']
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
menor valor 0.0 com os termos ['dourado', 'baixada', 'rasqueado', 'cuiaba', 'cruzeiro', 'fortaleza', 'clube', 'de', 'gloria', 'e', 'tradicao',
'salve', 'o', 'tricolor', 'paulista', 'athletico!', 'mantendo', 'nosso', 'espirito', 'esportivo']
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
menor valor 0.0 com os termos ['dourado', 'baixada', 'rasqueado', 'cuiaba', 'cruzeiro', 'fortaleza', 'clube', 'de', 'gloria', 'e', 'tradicao',
'salve', 'o', 'tricolor', 'paulista', 'athletico!', 'mantendo', 'nosso', 'espirito', 'esportivo']
doc doc1.txt tf*idf é :
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
menor valor 0.0 com os termos ['dourado', 'baixada', 'rasqueado', 'cuiaba', 'cruzeiro', 'fortaleza', 'clube', 'de', 'gloria', 'e', 'tradicao',
'salve', 'o', 'tricolor', 'paulista', 'athletico!', 'mantendo', 'nosso', 'espirito', 'esportivo']
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
menor valor 0.0 com os termos ['dourado', 'baixada', 'rasqueado', 'cuiaba', 'cruzeiro', 'fortaleza', 'clube', 'de', 'gloria', 'e', 'tradicao',
'salve', 'o', 'tricolor', 'paulista', 'athletico!', 'mantendo', 'nosso', 'espirito', 'esportivo']
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
menor valor 0.0 com os termos ['dourado', 'baixada', 'rasqueado', 'cuiaba', 'cruzeiro', 'fortaleza', 'clube', 'de', 'gloria', 'e', 'tradicao',
'salve', 'o', 'tricolor', 'paulista', 'athletico!', 'mantendo', 'nosso', 'espirito', 'esportivo']
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
menor valor 0.0 com os termos ['dourado', 'baixada', 'rasqueado', 'cuiaba', 'cruzeiro', 'fortaleza', 'clube', 'de', 'gloria', 'e', 'tradicao',
'salve', 'o', 'tricolor', 'paulista', 'athletico!', 'mantendo', 'nosso', 'espirito', 'esportivo']
doc doc2.txt tf*idf é :
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
```